Chapitre 2 : Organisation et Stockage des données

I. Stockage de données

I.1. Définition

Le stockage de données désigne l'ensemble des méthodes et technologies permettant de conserver des données numériques.
Les systèmes de stockage sont un ensemble d'équipements informatiques (ordinateur, connexion réseau, médias de stockage) et de logiciels appropriés, responsables du stockage à long terme de grandes masses d'informations et de leur accès.

I.2. Objectifs du stockage de données

Les solutions de stockage de données doivent répondre à un certain nombre d'impératifs. Si, la solution parfaite n'existe pas, ceux-ci constituent par contre des critères qu'il est important de prendre en compte pour choisir le meilleur compromis :

- **Intégrité** : La solution choisie devra permettre d'offrir une durée de vie théoriquement infinie aux données. On dit couramment que "les données doivent survivre aux incidents matériels".
- **Sécurité** : La solution doit permettre d'assurer la pérennité des données. On doit savoir contrôler et tracer l'ensemble des accès aux données et les niveaux de droits associés (lecture, écriture, création, etc.).
- **Performance** : L'architecture du modèle choisi doit être dimensionnée de manière cohérente avec les performances attendues (débit, temps de latence, etc.). Ce critère est particulièrement important lorsque l'on parle de stockage réseau : il faut alors s'interroger sur les investissements nécessaires en terme d'infra-structure.
- **Transparence** : L'utilisateur accédant aux données ne doit théoriquement pas savoir où et comment les données sont stockées, ni quels sont les accès concurrents au sien.
- **Limitation des coûts** : Pour des raisons évidentes, une bonne architecture de stockage doit permettre le stockage de grands volumes pour des coûts de mise en place et de maintenance les plus réduits possibles.

Afin d'assurer la persistance des données, les bases de données sont sauvegardées sous la forme de fichiers structurés en enregistrements sur une mémoire ou support de stockage. Ce support est le plus souvent un disque magnétique. C'est le niveau physique de l'architecture ANSI/SPARC qui gère et définit les formats ainsi que les méthodes d'accès des données. Le temps de lecture des données à partir d'un disque est largement plus grand que celui à partir de la mémoire principale. Cette dernière étant plus coûteuse, c'est l'optimisation de l'organisation des données sur un disque, les structures d'indexation et les algorithmes de recherche utilisés qui ont un rôle important dans l'amélioration des performances des SGBD.

I.3. Les mémoires

Une mémoire est un dispositif permettant de stocker une donnée et la maintenir dans le temps pour être accédée ultérieurement. Elle peut être volatile et perd son contenu à l'arrêt du système ou persistante et peut garder son contenu en absence d'alimentation.

I.3.1. Hiérarchie des mémoires

Les mémoires sont classées sous la forme d'une hiérarchie allant de la mémoire la plus petite et la plus rapide vers la plus volumineuse et plus lente. La hiérarchie des mémoires est représentée par la figure ci-dessous.

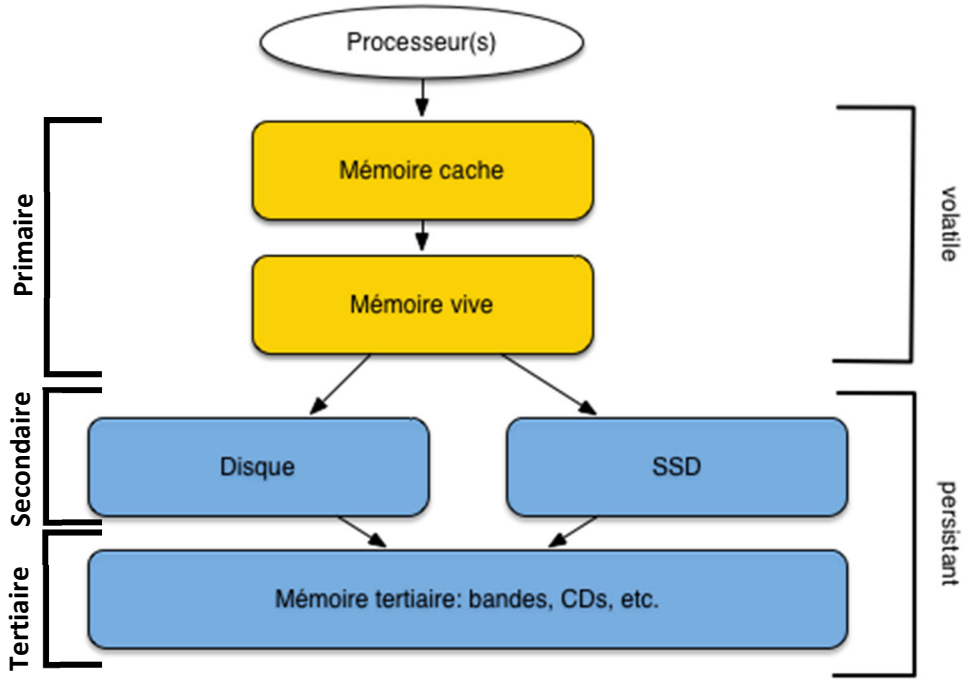


Figure 1. Hiérarchie des mémoires

I.3.1.1. Mémoire primaire

Cette catégorie comprend les mémoires qui peuvent être utilisées directement par l'unité centrale de traitement de l'ordinateur (CPU), comme la mémoire principale de l'ordinateur et la mémoire cache.

- La mémoire cache ou Static RAM est plus petite (en MO) mais plus rapide ; elle est utilisée par la CPU pour accélérer l'exécution des instructions d'un programme par le biais des techniques comme le prefetching et le pipelining.
- La mémoire principale ou mémoire vive (Dynamic RAM) est la zone de travail principale pour la CPU où sont chargées les instructions et les données d'un programme. Elle est moins chère que la mémoire cache mais également moins rapide.

Le stockage primaire permet généralement un accès rapide aux données, mais sa capacité de stockage est limitée. Bien que les capacités de la mémoire principale aient augmenté rapidement ces dernières années, les ans, ils sont toujours plus chers et ont une capacité de stockage inférieure à celle exigée par les bases de données. Le contenu de la mémoire principale est perdu en cas de coupure de courant ou de panne du système.

I.3.1.2. Mémoire secondaire

Le premier choix de support de stockage pour le stockage en ligne des bases de données d'entreprise a été les disques magnétiques. Cependant, les mémoires flash deviennent un support de choix pour le stockage de données à quantités modérées de données comme les SSD (Solid State Drive).

- Mémoires Flash : Ce sont des mémoires basées sur la technologie EEPROM (Electrically Erasable Programmable ROM). Elles offrent de bonnes performance ainsi qu'une grande densité mais chaque bloc doit être effacé puis écrit simultanément. Le forme la plus utilisée est celle des clés USB puis les cartes SD utilisées dans les appareils mobiles.
- Disques magnétiques : La catégorie la plus répandue de cette famille est les disques durs (HDD). Ils possèdent une surface rigide ou souple magnétisable par une tête de lecture/écriture.
- Disques SSD : Ils sont basés sur la mémoire flash et sont donc dépourvus d'éléments mobiles. Ils sont plus rapide mais une limite quant au nombre d'écritures possibles.

I.3.1.3. Mémoire tertiaire

Les disques optiques (CD-ROM, DVD et autres supports de stockage similaires) et les bandes magnétiques sont des supports amovibles utilisés dans les systèmes actuels en tant supports d'archivage de bases de données. Ces dispositifs ont généralement une plus grande capacité, coûtent moins cher et permettent un accès plus lent aux données que les dispositifs de stockage primaire.

- Disques optiques : Allant du CD au Blu-ray, ces disques sont très proche des disques magnétiques mais c'est le laser qui remplace la magnétisation d'où le nom optique.
- Bandes magnétiques : C'est le support d'archivage offrant la plus grande capacité de stockage et un temps d'accès le plus lent.

Les données stockées sur un support secondaire ou tertiaire ne peuvent pas être traitées directement par l'unité centrale ; d'abord, elles doivent être copiées dans la mémoire principale et ensuite traitées par l'unité centrale.

I.3.2. Performances des mémoires

Pour mesurer **les performances d'une mémoire**, deux critères essentiels sont à retenir :

- **Temps d'accès** : c'est le temps nécessaire pour aller à l'emplacement mémoire indiqué par une adresse mémoire et obtenir l'information. C'est un critère important quand on effectue des accès dits aléatoires.
- **Débit/Taux de transfert** : c'est le volume de données lues par unité de temps dans le meilleur des cas. Ce critère est important pour les accès dits séquentiels dans lesquels on lit une collection d'information, dans un ordre donné.

Le tableau suivant résumé les ordres de grandeur des temps d'accès pour les différentes mémoires.

Type	Ordre de taille	Temps d'accès (S)	Débit (par S)
Mémoire cache	MO	10^{-8}	10x GO
DRAM	GO	10^{-8} –10^{-7}	x GO
Disque magnétique	TO	10^{-2}	100 MO
SSD	TO	10^{-4}	x GO

I.4. Les supports de stockage

I.4.1. Les disques magnétiques

Les disques magnétiques constituent le principal périphérique de mémoire persistante ; ils offrent une grande capacité de stockage tout en gardant des accès en lecture et en écriture relativement efficaces.

Le schéma suivant décrit le fonctionnement d'un disque magnétique :

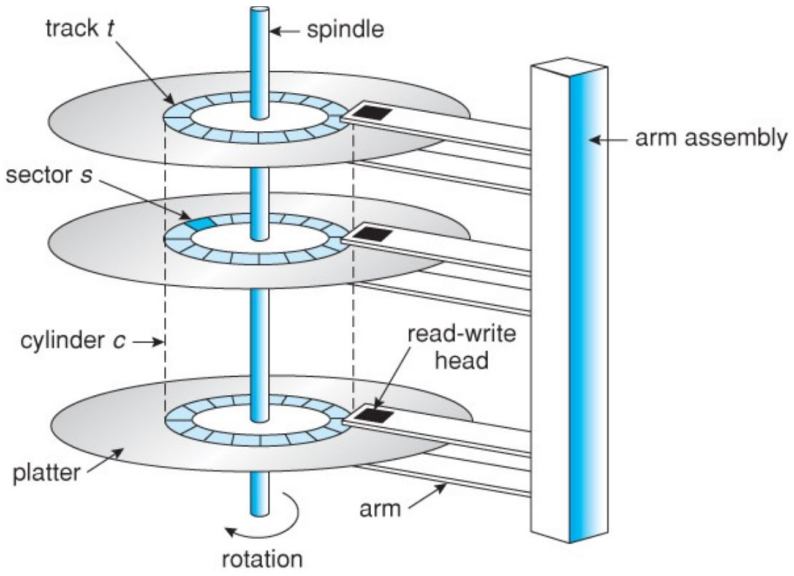


Figure 2. Éléments d'un disque magnétique

Un disque magnétique se compose principalement d'une surface magnétique rotative et d'un bras mécanique qui se déplace sur celle-ci. Le bras mécanique est utilisé pour lire et écrire sur le disque. Les données sur un disque magnétique sont lues et écrites par un processus de magnétisation. Le disque se compose de plusieurs plateaux ayant chacun sa propre tête de lecture/écriture. Les données sont organisées sur chaque plateau sous forme de pistes et de secteurs, où les pistes sont les divisions circulaires du disque. Les pistes sont en outre divisées en secteurs qui contiennent des blocs de données. Le formatage groupe les secteurs (de taille fixe 512O) en blocs de tailles égales (unité d'allocation 512O-8192O). Le bloc est l'unité d'E/S entre la mémoire principale et la mémoire secondaire. L'ensemble des pistes se situant au même niveau de chaque plateau constitue un cylindre virtuel.

Le temps d'accès se décompose en :

1. Délai de positionnement de la tête de lecture sur la piste contenant le bloc ;
2. Délai de latence : délai de rotation du disque pour attendre que le bloc passe sous la tête de lecture (rappelons que les têtes sont fixes, c'est le disque qui tourne) ;
3. Temps de transfert du bloc.

Pour minimiser ce temps, les blocs à lire/écrire doivent être placés dans le même secteur puis dans le même cylindre puis dans le cylindre adjacent.
Le tableau ci-dessous donne les spécifications d'un disque, telles qu'on peut les trouver sur le site de n'importe quel constructeur. Les chiffres donnent un ordre de grandeur pour les performances d'un disque, étant bien entendu que les disques destinés aux serveurs sont beaucoup plus performants que ceux destinés aux ordinateurs personnels.

Le modèle donné en exemple appartient au milieu de gamme :

Caractéristique	Performance
Capacité	2,7 To
Taux de transfert	100 Mo/s
Cache	3 Mo
Nbre de disques	3
Nbre de têtes	6
Nombre de cylindres	15 300
Vitesse de rotation	10 000 rpm (rotations par minute)
Délai de latence	En moyenne 3 ms
Temps de positionnement moyen	5,2 ms
Déplacement de piste à piste	0,6 ms

I.4.2. Les Solid State Drives (SSD)

Un disque Solid-State, ou SSD, est bâti sur la mémoire dite flash, celle utilisée pour les clés USB. Ce matériel est constitué de mémoires à semi-conducteurs à l'état solide. Contrairement aux disques magnétiques à rotation, les emplacements mémoire sont à accès direct, ce qui élimine le temps de latence.
Le temps d'accès direct est considérablement diminué, de l'ordre de quelques dixièmes de millisecondes, soit cent fois moins (encore une fois il s'agit d'un ordre de grandeur) que pour un disque magnétique.
Le débit en lecture/écriture est également bien plus important, de l'ordre de 1 Go par sec, soit 10 fois plus efficace. La conclusion simple est que le meilleur moyen d'améliorer les performances d'une base de données est de la placer sur un disque SSD.

I.5. Les technologies de stockage

I.5.1. Le RAID

La technologie RAID (pour *Redundant Array of Inexpensive Disks*) permet de combiner plusieurs disques durs en une seule unité de stockage virtuelle. Ainsi, une partie de l'espace est souvent réservée pour faire de la redondance d'informations. Les avantages de cette technologie sont multiples : souplesse du stockage, tolérance aux pannes et performance.
Il existe plusieurs configurations RAID comme :

- **RAID 0 :**

Le RAID 0, également connu sous le nom d'entrelacement de disques ou de "volumes agrégés par bandes" est une configuration RAID permettant d'augmenter les performances de la grappe en concaténant n disques durs qui fonctionneront ensemble pour constituer un volume plus large. On répartit les accès sur plusieurs disques, on accélère donc significativement les temps d'accès. Le RAID 0 n'apportant pas de redondance (donc pas de sécurité supplémentaire), tout l'espace disque disponible est utile. Le volume ainsi créé est donc moins fiable qu'un seul disque dur : la perte d'un seul disque entraîne la perte de toutes les données.

- **RAID 1 :**

Le RAID 1 consiste en l'utilisation de plusieurs disques redondants, chaque disque de la grappe contenant à tout moment exactement les mêmes données : on parle aussi de miroir.
La capacité totale est égale à celle du plus petit élément de la grappe : il est donc conseillé d'utiliser des éléments identiques. Cette solution offre un excellent niveau de protection des données. Elle accepte une défaillance de n-1 éléments (ou n est le nombre de disques) sans perte

de données. En cas de défaillance, les disques sont automatiquement désactivés, puis reconstruits après remplacement de manière transparente pour les utilisateurs.
La contrepartie est le coût très élevé de cette solution. Plus le nombre de miroirs est élevé, et plus la sécurité augmente, mais plus son coût devient prohibitif.

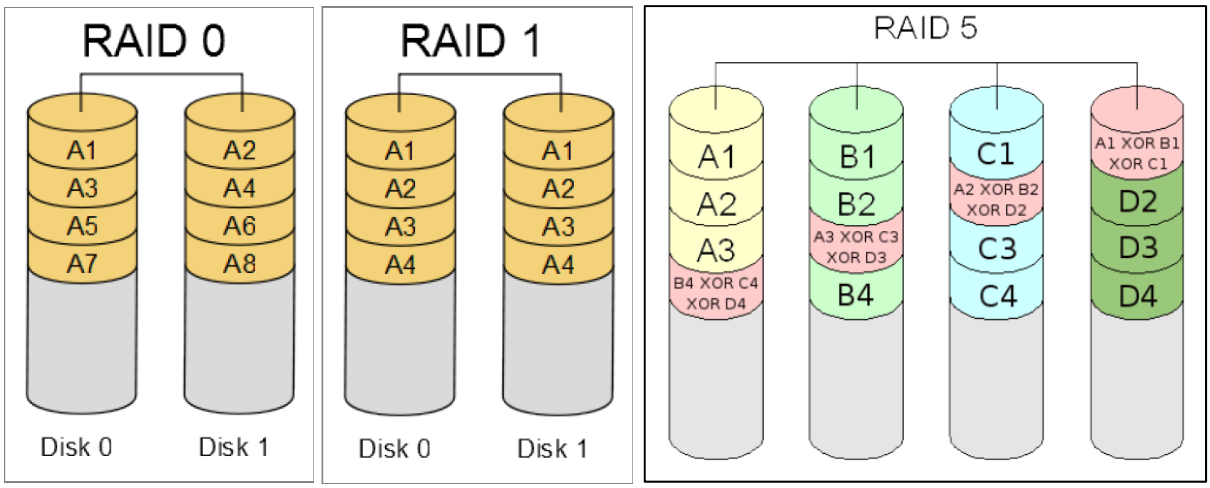


Figure 3. Technologie RAID

- **RAID 5 :**

Le RAID 5 combine la méthode du volume agrégé et de la parité. Il s'agit donc d'un compromis permettant d'allier performance et sécurité pour un coût moindre.
La parité, qui est incluse avec chaque écriture se retrouve répartie circulairement sur les différents disques.
L'intégrité des données de chaque bande est donc préservée : non seulement la grappe est toujours en état de fonctionner, mais il est de plus possible de reconstruire le disque une fois échangé à partir des données et des informations de parité contenues sur les autres disques.
RAID 5 supporte donc la perte d'un seul disque.
Ce système nécessite impérativement un minimum de trois disques durs, toutefois on considère généralement que les meilleures performances sont obtenues pour 5, 9 et 14 disques. Ceux-ci doivent théoriquement être de même taille.
La capacité de stockage utile réelle, pour un système de n disques dont le plus petit à une capacité c identiques est de (n - 1) * c.

I.5.2. SCSI

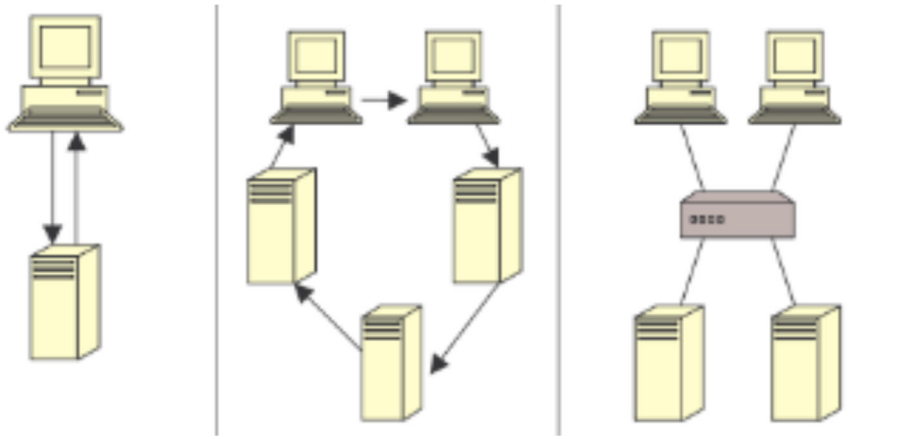
La technologie SCSI (pour *Small Computer System Interface*) est un standard de bus informatique permettant de relier des ordinateurs et des périphériques très utilisé par les périphériques de stockage. Elle fournit une interface parallèle avec un débit allant jusqu'à 640 Mo/s dans ses dernières spécifications. La longueur des câbles peut aller jusque 20 m., et reste donc limitante. Enfin, la seule topologie gérée est logiquement le point à point.
De part son fonctionnement, SCSI est une interface plus rapide, mais aussi plus complexe que d'autres interfaces telles qu'E-IDE : elle repose notamment sur le fait que les équipements reliés sauront recevoir des instructions complexes. On lui préfère aujourd'hui des liaisons Fibre Channel.

I.5.3. Fibre Channel

Initialement conçues pour les super-ordinateurs, les technologies de fibre optiques sont aujourd'hui devenu un standard dans les réseaux de stockage.

Il convient lorsque l'on parle de fibre channel de bien faire la différence entre le protocole et le support qui peuvent être utilisés indépendamment.
Le protocole FC définit 3 topologies distinctes.

- **Point à point** : les périphériques sont reliés directement entre eux. La bande passante est entièrement dédiée aux échanges entre ces 2 machines.
- **En boucle** (aussi appelé Arbitred Loop ou FC-AL) : dans ce cas tous les périphériques sont reliés. On peut avoir jusqu'à 126 périphériques ainsi reliés. La boucle est gérée par des éléments hubs.
- **Switchée ou commutée (aussi appelé Fabric)** : on utilise dans ce cas un switch Fibre Channel dont le principe de fonctionnement est le même que pour les échanges Ethernet. Dans cette configuration, on rencontre un maximum théorique de 16 millions de composants.



Point à point **En boucle** **Switchée ou commutée**

Figure 4. Topologies Fibre Channel

Ce protocole est aujourd'hui le plus fréquemment utilisé sur des supports cuivre ou fibre optique. La fibre optique est utilisée dans les réseaux de stockage pour le transfert de données, mais aussi pour l'échange d'informations.

II. Organisation des fichiers :

II.1. Fichier, bloc et enregistrement

Les données gérées par un SGBD sont sauvegardées dans des fichiers comme le cas des systèmes d'exploitation. Ces fichiers ont une structure spécifique : ils sont composés de blocs (ou page), où chaque bloc contient des enregistrements (Record). Dans le cas des SGBD relationnels, nous pouvons assimiler une relation à un fichier, et chaque ligne (row) lui correspond un enregistrement. Les enregistrements contiennent les valeurs des différents champs (field) décrivant une ligne. Ces derniers représentent l'unité de haut niveau pour les opérations de lecture/écriture d'un SGBD. Ils doivent, donc, être placés sur disque de manière à les localiser assez rapidement. La stratégie adoptée pour placer les enregistrements est définie par l'organisation du fichier dont l'objectif d'optimiser l'utilisation de l'espace disque (allocation des blocs) et de minimiser le temps des E/S.

II.1.1. Enregistrement

Selon le type des champs d'un enregistrement il peut être de taille fixe ou de taille variable (type Varchar). En général, la taille de l'enregistrement est très inférieure à celle du bloc.

Quand un champ est de taille variable, le SGBD doit déterminer la taille utile. Ceci peut être réalisé soit en préfixant la valeur par sa taille ou en terminant la valeur par une valeur spéciale.

Exemple : un champ de type varchar(20) qui contient la chaine 'Bonjour' peut être représenté par : 7'Bonjour' dans le premier cas ou 'Bonjour$' si $ est le caractère de fin du champ.

Enregistrement à taille fixe : la structure et la taille de chaque champ sont définies dans le schéma de la table (catalogue de la base de données). Pour lire l'enregistrement d'ordre *n* il suffit d'effectuer un déplacement de *n* x taille de l'enregistrement. Ensuite récupérer les valeurs de chaque champ selon leurs tailles respectives. La figure ci-dessous représente la structure d'un enregistrement à taille fixe.

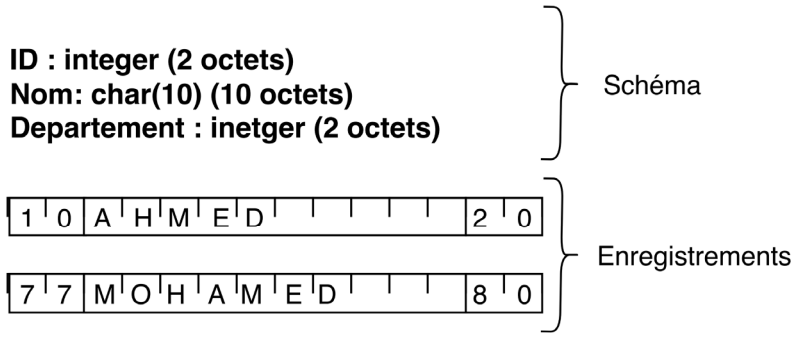


Figure 5. Enregistrement à taille fixe

Enregistrement à taille variable : Comme pour le cas des champs à taille variable, des informations complémentaires sont stockées dans l'entête de chaque enregistrement tel que :

- La taille de l'enregistrement,
- Pointeur vers le schéma pour déterminer sont type,
- Masque pour les valeurs NULL,
- Date de dernière mise à jour,
- Etc.

Exemple : Prenons l'exemple d'une table Film avec les attributs id de type INTEGER (4 octets), titre de type VARCHAR(50) et annee de type INTEGER. Regardons la représentation de l'enregistrement (123, 'Vertigo', NULL) (donc l'année est inconnue).

L'identifiant est stocké sur 4 octets, et le titre sur 8 octets, dont un pour la longueur. L'entête de l'enregistrement contient un pointeur vers le schéma de la table, sa longueur totale (soit 4 + 8), et un masque de bits 110 indiquant que le troisième champ est à NULL.

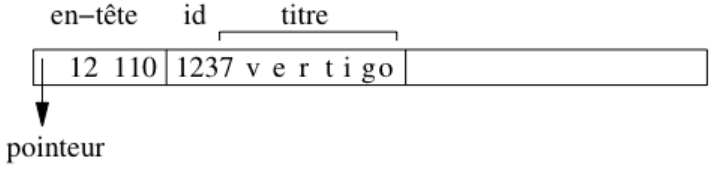


Figure 6. Enregistrement à taille variable

II.1.2. Bloc et allocation

Les enregistrements sont affectés à un bloc qui est l'unité de transfert entre le disque et la mémoire. Pour un bloc de taille B et un enregistrement de taille E, le nombre maximal d'enregistrement par bloc est égal à $\left\lfloor \frac{B}{E} \right\rfloor$ appelé *bfr* (Blocking factor).

Plusieurs options d'allocation de blocs sont possibles pour :

- Séparer les enregistrements (inutile pour les enregistrements de taille fixe) : ajouter un marqueur de fin d'enregistrement ou mettre la taille de chaque enregistrement dans l'entête du bloc.
- Fractionner les enregistrements entre blocs : si les enregistrements ne sont pas fractionnés (figure 7(a)) alors il y aura une perte d'un espace de stockage égal à $B - bfr * E$. Fractionner les enregistrements permet d'utiliser cet espace et devient une obligation quand la taille de l'enregistrement est supérieure à celle du bloc (figure 7(b)). Dans ce cas, il faut prévoir un pointeur vers la suite de l'enregistrement.

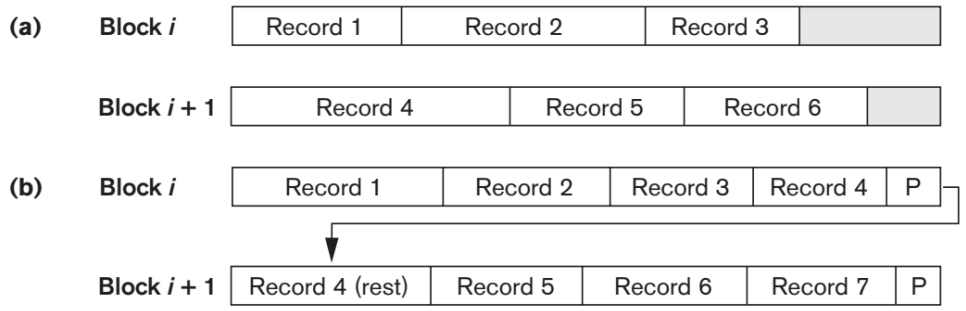


Figure 7. Fractionnement des enregistrements

- Allocation de blocs sur disque : une première option est l'allocation contiguë i.e. les blocs de fichiers sont placés dans des blocs de disque adjacents ce qui accélère la lecture mais pose des problèmes lors de l'augmentation de taille du fichier. Un autre choix est l'allocation avec chaînage où chaque bloc contient un pointeur vers le bloc suivant.

Un bloc est structuré en un entête suivi de la zone réservée pour les enregistrements (ensemble de cellules). L'entête contient un répertoire de pointeurs vers les différents enregistrements du bloc ainsi que le chaînage vers le bloc ou la page suivante.

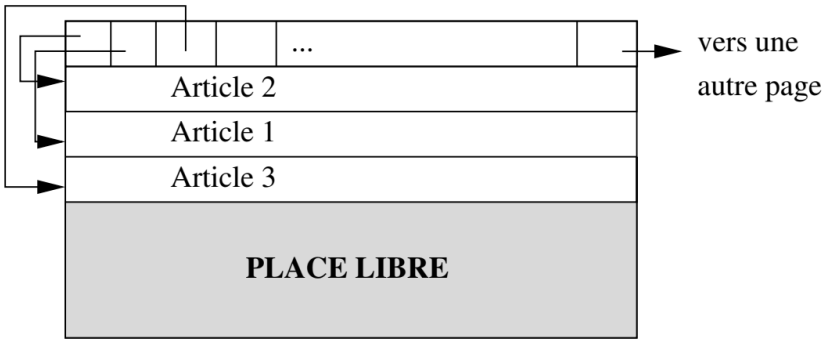


Figure 8. Structure générale d'un bloc

Quand les enregistrements sont de taille fixe, l'entête est simple et peut être réduite aux information de nombre de cellules et masque indiquant leurs états (libre ou occupé).

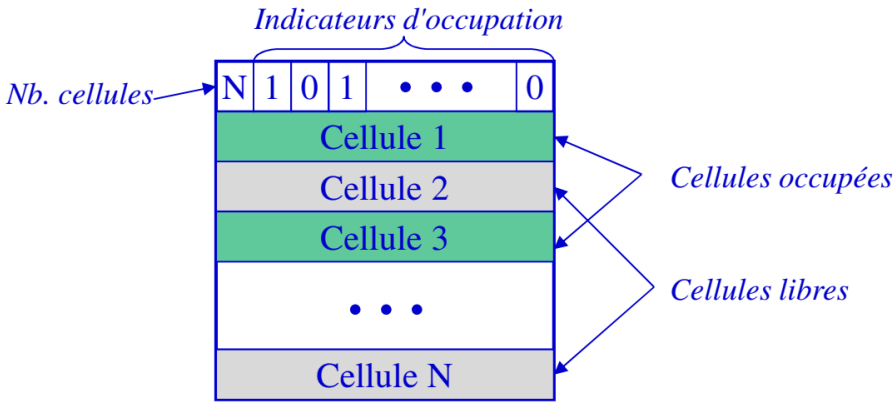


Figure 9. Entête de bloc avec enregistrement à taille fixe

Exemple : Supposons que la taille de l'enregistrement est de 84 octets, et que la taille de bloc est de 4096 octets. Chaque bloc contient un en-tête de 100 octets pour stocker des informations comme l'espace libre disponible dans le bloc, un chaînage avec d'autres blocs, etc. On désire lire l'enregistrement numéro 563. Pour cela il faut déterminer le numéro de bloc et le numéro de l'enregistrement dans le bloc.

Le nombre d'enregistrement par bloc est de : $\left\lfloor \frac{4096-100}{84} \right\rfloor = 47$

Le numéro de bloc est : $\left\lfloor \frac{563}{47} \right\rfloor + 1 = 12$

Le numéro du bloc dans l'enregistrement est : $563 - (12 - 1) \times 47) = 46$

II.2. Organisation primaire

Elle décrit comment les enregistrements sont physiquement écrit sur disque. On distingue les types suivants :

- Fichier séquentiel non ordonné (heap file) : c'est la structure la plus simple ; les enregistrements sont placés sans aucun ordre par l'ajout à la fin du fichier. Avec la croissance/diminution de la taille du fichier, les blocs du disque sont alloués ou libérés. Pour gérer ces fichiers, un SGBD doit enregistrer les blocs d'un fichier, l'espace libre dans les blocs et les enregistrements d'un bloc. Un exemple est donné dans la figure suivante :

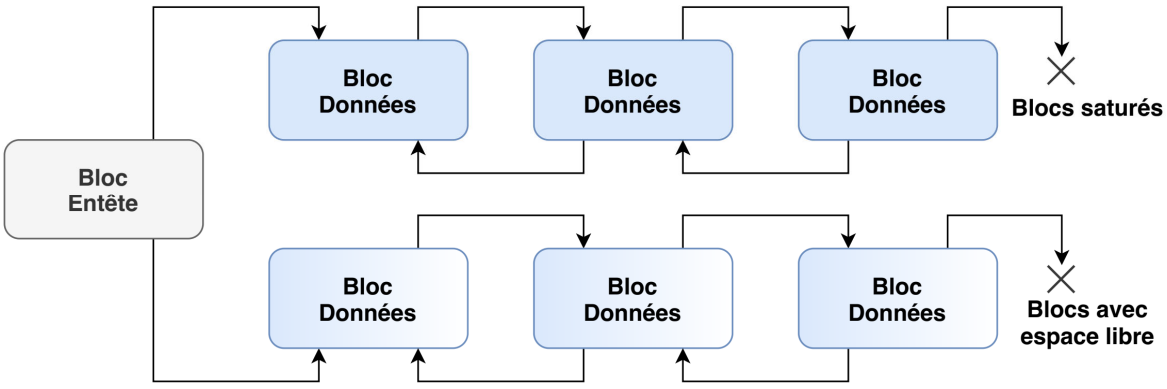


Figure 10. Exemple de structure de fichier non ordonné

L'insertion est moins coûteuse mais la recherche, la suppression et la mise à jour sont moins performants : en moyenne $\left\lceil \frac{b}{2} \right\rceil$ blocs sont accédés et au pire des cas b blocs si le fichier contient b blocs.

- Fichier séquentiel trié : les enregistrements sont ordonnés selon la valeur d'un champ particulier. Ceci permet un accès rapide sur le champ de tri qui a la même complexité qu'une recherche dichotomique ($log_2 b$). Par contre, l'ajout est plus coûteux car il requiert des opérations de décalage d'enregistrements.
- Fichier haché : les enregistrements ne sont pas triés et sont placés dans un ordre défini par une fonction de hachage appliqué sur la valeur d'un champ de l'enregistrement (clé de hachage). Par exemple, soit *x* est la valeur du champ du nouvel enregistrement et *r* le nombre d'enregistrements du fichier. Pour trouver la position du nouvel enregistrement, on applique *x* mod *r*. Si la position calculée est déjà occupée on parle de collision. Elle peut être résolue en cherchant la première position libre ou par l'application d'une seconde fonction de hachage ou encore par placer l'enregistrement dans une zone de débordement.

Exemple : On dispose de 5 fragments (f_0-f_4) contenant chacun un bloc de capacité 4 enregistrements représentant un film. La fonction de hachage sur le titre du film permet d'associer chaque film à un fragment. Elle doit donc retourner un entier de 0 à 4 avec une probabilité uniforme. Un exemple possible est ***h*(*titre*)=*rang*(*titre*[0]) *mod* 5**. C'est à dire appliquer la fonction modulo 5 au premier caractère du titre. Dans le cas du film « Citizen kane », rang(c) mod 5 = 3 mod 5 = 3 ➔f_3. Puisque ce fragment est plein alors une nouvelle de zone de débordement est créée et chaînée au fragment initial pour pouvoir y insérer ce film.

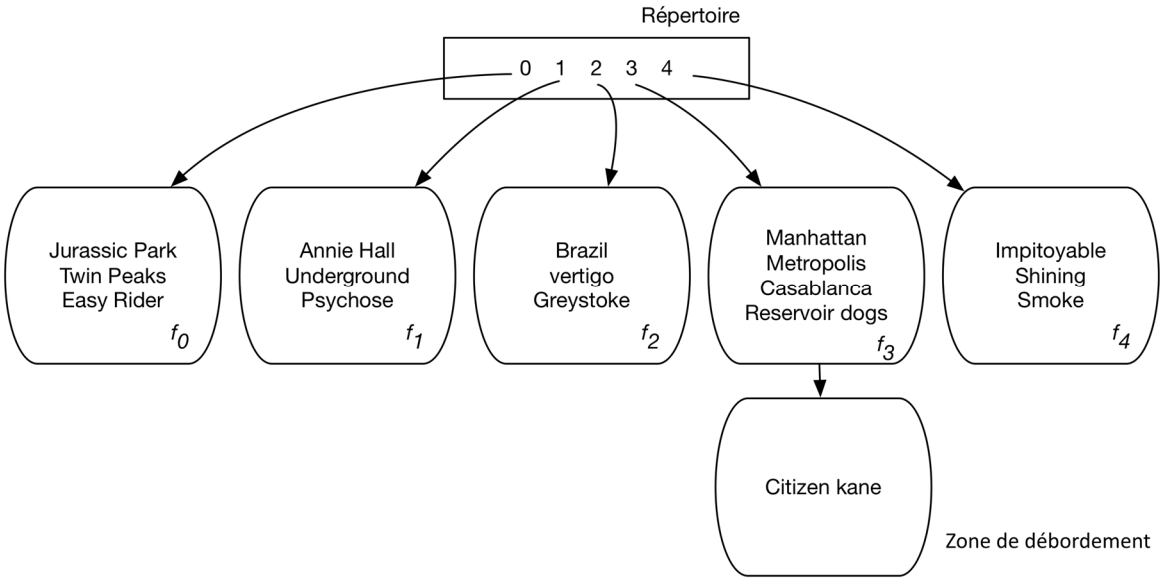


Figure 11. Hachage statique et zone de débordement

- Autres structures : comme les fichiers à structure d'arbre. La structure la plus utilisée est celle des arbres B (B-tree). C'est une généralisation des arbres binaires de recherche permettant de garder un arbre équilibré et ordonnée avec des temps de recherche, insertion et suppression logarithmiques.

II.3. Organisation secondaire

Connue aussi sous le nom de structure d'accès auxiliaire permettant un accès efficace aux enregistrements d'un fichier en se basant sur des champs alternatifs autres que ceux utilisés

dans l'organisation primaire. Ils sont souvent sous forme d'index sur les clés. Avec cette organisation 2 fichiers sont utilisés : le fichier de données et le fichier d'index.

Un index (comme celui dans les livres) est un fichier contenant les clés et les adresses correspondantes. Si l'index contient toutes les valeurs de la clé, il est qualifié de dense. Les index peuvent être à 1-niveau, multi-niveaux ou encore multi-niveau dynamiques.

II.3.1. Index à 1-niveau

Index primaire :

Le fichier de donnée étant trié selon une clé, l'index primaire est trié aussi et contient des enregistrements permettant d'associer à une clé l'adresse d'un bloc. Si une clé *c* est associée à l'adresse *a* alors le premier enregistrement du bloc *a* possède la clé *c*. Un exemple est illustré par la figure suivante :

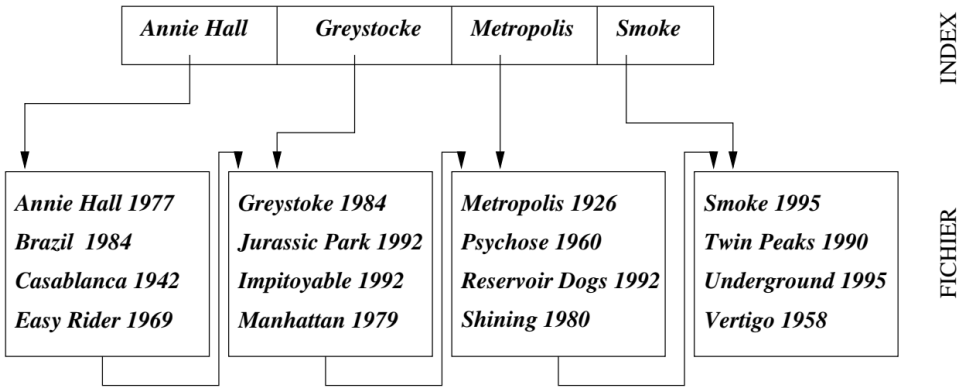


Figure 12. Index primaire

L'index primaire est non dense. Le nombre d'enregistrements de l'index est très inférieur à celui du fichier de données (égal au nombre de blocs du fichier de données). La taille d'un enregistrement du fichier d'index est petite ce qui permet un nombre important d'enregistrement par bloc. C'est ce qui rend la recherche dans l'index primaire plus rapide.

Index clustérisé :

Le champ de tri n'est pas une clé et par la suite il est possible d'avoir des doublons. Dans ce cas, le fichier d'un index contient un enregistrement par valeur distincte du champ de tri. À cette dernière est associé un pointeur vers le premier bloc dans lequel cette valeur apparaît.

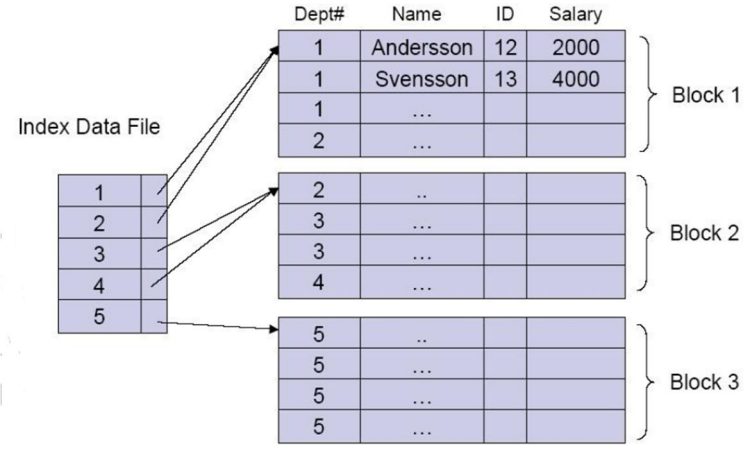


Figure 13. Index clustérisé

Index secondaire :

Avec ce type d'index, le champ (clé ou non) utilisé n'est pas celui utilisé pour le tri. C'est un index dense qui est souvent utilisé.

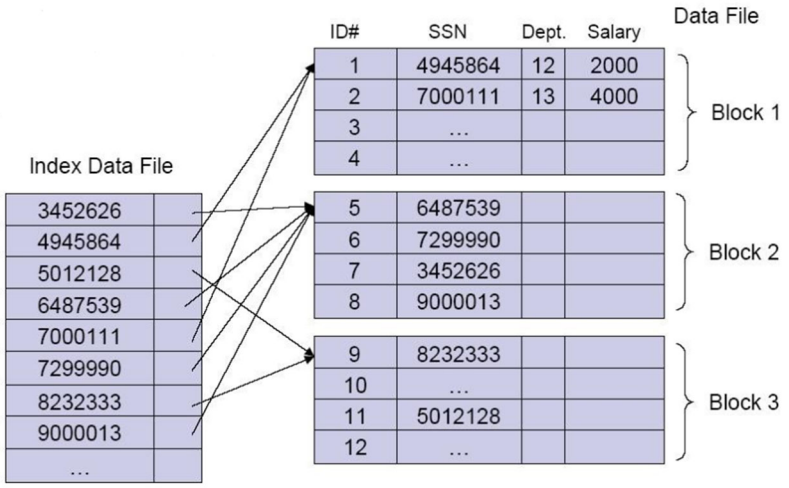


Figure 14. Index Secondaire

II.3.2. Index multi-niveau

Comme son nom l'indique, plusieurs niveaux d'indexation sont utilisés par l'ajour d'index sur les fichiers d'index eux-mêmes. Il peut être primaire, secondaire ou clustérisé. On ajoute des index jusqu'à ce que le dernier index peut être placé dans un unique bloc. Pour accéder à un enregistrement de données avec ce type d'index, il faut accéder à N+1 blocs où N est le nombre de niveaux.

III. Organisation des données dans Oracle

La structure globale de stockage d'une base de données oracle est composée d'une structure visible à l'OS et d'une structure invisible à l'OS. La structure visible à l'OS correspond à la structure physique c'est-à-dire les fichiers systèmes de base de données qui existent au niveaux du système d'exploitation et la structure invisible correspond à la structure logique de la base de données c'est-à-dire les tablespaces.

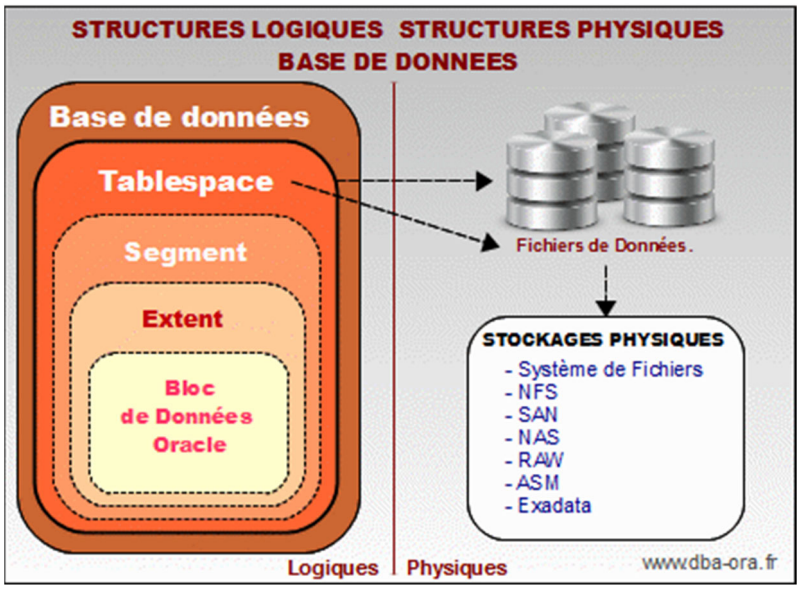


Figure 15. Organisation logique et physique dans Oracle

III.1. Structure logique

III.1.1. Bloc de données

Le bloc est la plus petite unité de stockage gérée par ORACLE. La taille d'un bloc peut être choisie au moment de l'initialisation d'une base, et correspond obligatoirement à un multiple de la taille des blocs du système d'exploitation. À titre d'exemple, un bloc dans un système comme Linux occupe 1024 octets, et un bloc ORACLE occupe typiquement 4 096 ou 8 092 octets.

La structure d'un bloc est identique quel que soit le type d'information qui y est stocké. Elle est constituée des cinq parties suivantes :

- l'entête (header) contient l'adresse du bloc, et son type (données, index, etc) ;
- le répertoire des tables donne la liste des tables pour lesquelles des informations sont stockées dans e bloc ;
- le répertoire des enregistrements contient les adresses des enregistrements du bloc ;
- un espace libre est laissé pour faciliter l'insertion de nouveaux enregistrements, ou l'agrandissement des enregistrements du bloc (par exemple un attribut à NULL auquel on donne une valeur par un update).
- enfin, l'espace des données contient les enregistrements.

La taille d'un bloc est spécifiée grâce au paramètre **DB_BLOCK_SIZE** dans le fichier **init.ora**.

L'espace libre dans un bloc est géré par les paramètres **pctfree et pctused.**

Par exemple : Si pctfree vaut 30%, et pctused 40% (notez que la somme de ces deux valeurs ne peut jamais excéder 100%), les insertions dans un bloc peuvent donc s'effectuer jusqu'à ce que 70% du bloc soit occupé. Le bloc est alors retiré de la liste des blocs disponibles pour des insertions, et seules des mises à jour (destructions ou modifications) peuvent affecter son contenu. Si, à la suite de ces mises à jour, l'espace occupé tombe en-dessous de 40%, le bloc est à nouveau marqué comme étant disponible pour des insertions.

III.1.2. Extent (Extension)

Une extension est une suite contiguë (au sens de l'emplacement sur le disque) de blocs. En général une extension est affectée à un seul type de données (par exemple les enregistrements d'une table). Cette contiguïté est un facteur essentiel pour l'efficacité de l'accès aux données, puisqu'elle évite les déplacements des têtes de lecture, ainsi que le délai de rotation.
Le nombre de blocs dans une extension peut être spécifié par l'administrateur.

III.1.3. Segment

Un segment est un ensemble de fragments de stockage pour un des types de données persistantes géré par Oracle.
Il existe de nombreux types de segments, voici les principaux :

- **les segments de données** contiennent les enregistrements des tables, avec un segment de ce type par table ;
- **les segments d'index** contiennent les enregistrements des index ; il y a un segment par index ;
- **les segments temporaires** sont utilisés pour stocker des données pendant l'exécution des requêtes (par exemple pour les tris) ;

- **les segments rollbacks** contiennent les informations permettant d'effectuer une reprise sur panne ou l'annulation d'une transaction ; il s'agit typiquement des données avant modification, dans une transaction qui n'a pas encore été validée.

III.1.4. Tablespace

Un tablespace est un espace physique constitué d'un ou plusieurs fichiers. Une base de données Oracle est donc organisée sous la forme d'un ensemble de tablespace, sachant qu'il en existe toujours un, créé au moment de l'initialisation de la base, et nommé SYSTEM. Ce tablespace contient le dictionnaire de données, y compris les procédures stockées, les triggers, etc.
L'organisation du stockage au sein d'un tablespace est décrite par de nombreux paramètres (taille des extensions, nombre maximal d'extensions, etc.) qui sont donnés à la création du tablespace, et peuvent être modifiés par la suite. C'est donc au niveau du tablespace (et pas au niveau du fichier) que l'administrateur de la base peut décrire le mode de stockage des données.

La création de plusieurs tablespaces, avec des paramètres de stockage individualisés, offre de nombreuses possibilités :

- adaptation du mode de stockage en fonction d'un type de données particulier ;
- affectation d'un espace disque limité aux utilisateurs ;
- contrôle sur la disponibilité de parties de la base, par mise hors service d'un ou plusieurs tablespaces ;
- enfin – et surtout – répartition des données sur plusieurs disques afin d'améliorer les performances.

Au moment de la création d'un tablespace, on indique les paramètres de stockage par défaut des tables ou index qui seront stockés dans ce tablespace. L'expression "par défaut" signifie qu'il est possible, lors de la création d'une table particulière, de donner des paramètres spécifiques à cette table, mais que les paramètres du tablespace s'appliquent si on ne le fait pas.

Les principaux paramètres de stockage sont :

- la taille de l'extension initiale (par défaut 5 blocs) ;
- la taille de chaque nouvelle extension (par défaut 5 blocs également) ;
- le nombre maximal d'extensions, ce qui donne donc, avec la taille des extensions, le nombre maximal de blocs alloués à une table ou index ;
- la taille des extensions peut croître progressivement, selon un ratio indiqué par pctincrease; une valeur de 50% pour ce paramètre indique par exemple que chaque nouvelle extension a une taille supérieure de 50% à la précédente.

Voici un exemple de création de tablespace :

```
CREATE TABLESPACE TB1
DATAFILE 'fichierTB1.dat' SIZE 50M
DEFAULT STORAGE (
INITIAL 100K
NEXT 40K
MAXEXTENTS 20,
PCTINCREASE 20);
```

La commande crée un tablespace, nommé TB1, et lui affecte un premier fichier de 50 mégaoctets. Les paramètres de la partie DEFAULT STORAGE indiquent, dans l'ordre :

- la taille de la première extension allouée à une table (ou un index) ;
- la taille de la prochaine extension, si l'espace alloué à la table doit être agrandi ;
- le nombre maximal d'extensions, ici 20;
- enfin chaque nouvelle extension est 20% plus grande que la précédente.

En supposant que la taille d'un bloc est 4K, on obtient une première extension de 25 blocs, une seconde de 10 blocs, une troisième de 10 × 1, 2 = 12 blocs, etc.

Le fait d'indiquer une taille maximale permet de contrôler que l'espace ne sera pas utilisé sans limite, et sans contrôle de l'administrateur. En contrepartie, ce dernier doit être prêt à prendre des mesures pour répondre aux demandes des utilisateurs quand des messages sont produits par Oracle indiquant qu'une table a atteint sa taille limite.

Voici un exemple de tablespace défini avec un paramétrage plus souple : d'une part il n'y a pas de limite au nombre d'extensions d'une table, d'autre part le fichier est en mode auto-extension, ce qui signifie qu'il s'étend automatiquement, par tranches de 5 mégaoctets, au fur et à mesure que les besoins en espace augmentent. La taille du fichier est elle-même limitée à 500 mégaoctets :

```
CREATE TABLESPACE TB2
DATAFILE 'fichierTB2.dat' SIZE 2M
AUTOEXTEND ON NEXT 5M MAXSIZE 500M
DEFAULT STORAGE (INITIAL 128K NEXT 128K
MAXEXTENTS UNLIMITED);
```

Il est possible, après la création d'un tablespace, de modifier ses paramètres, étant entendu que la modification ne s'applique pas aux tables existantes mais à celles qui vont être créées. Par exemple on peut modifier le tablespace TB1 pour que les extensions soient de 100K, et le nombre maximal d'extensions porté à 200 :

```
ALTER TABLESPACE TB1
DEFAULT STORAGE (
NEXT 100K
MAXEXTENTS 200);
```

On peut mettre un tablespace hors-service, soit pour effectuer une sauvegarde d'une partie de la base, soit pour rendre cette partie de la base indisponible :

```
ALTER TABLESPACE TB1 OFFLINE;
```

On peut mettre un tablespace en lecture seule :

ALTER TABLESPACE TB1 **READ ONLY**;

On peut ajouter un nouveau fichier à un tablespace afin d'augmenter sa capacité de stockage :

ALTER TABLESPACE **ADD** DATAFILE 'fichierTB1-2.dat' **SIZE** 300 M;

Oracle fournit un certain nombre de vues dans son dictionnaire de données pour consulter l'organisation physique d'une base, et l'utilisation de l'espace :

- La vue **DBA_EXTENTS** donne la liste des extensions ;
- La vue **DBA_SEGMENTS** donne la liste des segments ;
- La vue **DBA_FREE_SPACE** permet de mesurer l'espace libre ;
- La vue **DBA_TABLESPACES** donne la liste des tablespaces ;
- La vue **DBA_DATA_FILES** donne la liste des fichiers.

Ces vues sont gérées sous le compte utilisateur SYS qui est réservé à l'administrateur de la base.

Il est possible, au moment où on spécifie le profil d'un utilisateur, d'indiquer dans quels tablespaces il a le droit de placer des tables, de quel espace total il dispose sur chacun de ces tablespaces, et quel est le tablespace par défaut pour cet utilisateur.

Il devient alors possible d'inclure dans la commande CREATE TABLE des paramètres de stockage. Voici un exemple, :

CREATE TABLE Film (...)
PCTFREE 10
PCTUSED 40
TABLESPACE TB1
STORAGE (INITIAL 50K
NEXT 50K
MAXEXTENTS 10
PCTINCREASE 25);

On indique donc que la table doit être stockée dans le tablespace TB1, et on remplace les paramètres de stockage de ce tablespace par des paramètres spécifiques à la table Film.

III.2. Structure physique

III.2.1. Fichiers de données

Chaque base de données Oracle possède **au minimum un fichier** de données. Ces fichiers contiennent **toutes les données de la base**. Les données des structures logiques comme les tables ou index sont stockées dans les fichiers de données.

III.2.2. Fichiers de contrôle

Ce sont des fichiers qui contiennent les données sur la base de données elle-même on les appelle les métadonnées. Ces fichiers sont critiques à la base de données, sans eux, il est impossible d'ouvrir les data files (fichiers de données) pour accéder aux données de la base de données.
Chaque base de données Oracle possède **un fichier de contrôle** ; le fichier de contrôle contient des **informations** sur la **structure physique** de la base. Par exemple, il référence des informations telle que le nom de la base, le nom et le chemin des fichiers de données et des fichiers de redo log.

III.2.3. Fichiers de journalisation (Fichiers redo log)

Ce sont des fichiers qui vont servir au recouvrement de la base de données en cas de crashe de celle-ci. Chaque base de données Oracle possède un ensemble d'**au moins deux fichiers redo log** : cet ensemble est nommé le redo log de la base. La fonction principale du redo log est d'**enregistrer toutes les modifications sur les données**. Si un problème devait empêcher les données d'être écrites de façon permanente sur le disque, les modi
fications seraient obtenues depuis le redo log. Le redo log étant **vital** pour le maintien de la base de données, Oracle permet de **multiplexer** le redo log ce qui signifie qu'une ou plusieurs copies du redo log peuvent être **maintenues sur des disques différents.**

III.2.4. Autres fichiers

- **Fichiers de paramètres** : Utilisé pour définir comment l'instance sera configurée à son démarrage.

- **Fichiers de mots de passe** : Permet aux utilisateurs de se connecter à distance sur la base de données et d'effectuer des tâches d'administrations.

- **Archives Log Files** : Ces fichiers contiennent l'historique en cours du redo générée par l'instance, ils permettent le recouvrement de la base de données. L'utilisation de ces fichiers avec le backup de la base de données permet de recouvrer les data files perdues.